VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

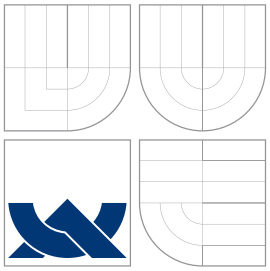
# MĚŘENÍ VZDÁLENOSTI POMOCÍ FOTOMOBILU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

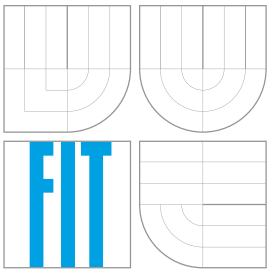
AUTOR PRÁCE
AUTHOR

KAREL POKORNÝ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# MĚŘENÍ VZDÁLENOSTI POMOCÍ FOTOMOBILU

DISTANCE MEASURING BY MOBILE PHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — KAREL POKORNÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZÖKE
SUPERVISOR

BRNO 2008

# Abstrakt

Cílem této práce je nalezení způsobu, jak naprogramovat aplikaci pro mobilní telefon, která bude schopná měřit vzdálenost, kterou přístroj urazí nad vhodným povrchem, a to s využitím integrované kamery mobilního telefonu. K tomu je nutné využít některou z metod detekce pohybu. Konečná aplikace používá algoritmus pro odhad konstatního optického toku, jehož výsledkem je jedinná hodnota optického toku mezi dvěma následujícími snímky - předpokladem je tedy pohyb celé scény v obraze. Aplikace byla realizována pro operační systém Symbian s uživatelským rozhraním S60 3rd Edition. Implementačním jazykem je C++.



# Klíčová slova

Optický tok, Konstantní optický tok, Detekce pohybu s využitím významných bodů, Symbian OS



# Abstract

Target of this thesis is to find out the way to create a mobile phone application, that is capable of meassuring the distance, phone covers above a suitable surface, using integrated camera. Usage of one of motion detection methods is needed to reach the target. Final application uses constant optical flow estimation algorithm. It gives only one value of optical flow between two succeding video frames as a result - movement of whole scene in the picture is assumed. Application was realized for Symbian operating system with S60 3rd Edition user interface. Implementation language is C++.



# Keywords

Optical flow, Constant optical flow, Motion detection based on correspondence of interest points, Symbian OS



# Citace

Karel Pokorný: Měření vzdálenosti pomocí fotomobilu, bakalářská práce, Brno, FIT VUT v Brně, 2008

# Měření vzdálenosti pomocí fotomobilu

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana ing. Igora Szökeho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Karel Pokorný
13. května 2008


## Poděkování

Děkuji mému vedoucímu, kterým byl pan ing. Igor Szöke, za trpělivost, vstřícnost, cenné rady a konstruktivní připomínky při realizaci práce. Děkuji také svým rodičům a přátelům za psychickou podporu.

Licenční smlouva a zadání jsou vloženy v originálním výtisku bakalářské práce.

# Obsah

# Kapitola 1

# Úvod

Mobilní telefony mají stále se zvětšující potenciál a dokáží zastoupit funkce velkého množství elektronických přístrojů. Výkon, kterým kdysi disponoval jen málokterý osobní počítač si dnes nosí každý v kapse. Dnešní mobilní telefon se však svými možnostmi osobnímu počítači minulosti nejen vyrovná, ale v určitých ohledech jej daleko předčí. Jednou z takových výhod je integrace digitálního fotoaparátu. Umožňuje nejenom fotografovat a natáčet video, ale skýtá i široké využití v oblasti zpracování digitálního obrazu.

Aplikace zpracovávající obraz však obvykle nejsou nedodávány výrobci mobilních telefonů. Jejich vývoj by nebyl možný bez existence platformy, která umožňuje programovat mobilní aplikace téměř každému. Takových platforem je celá řada, nejrozšířenější je Java MicroEdition, ale existují i další. Mnohostranné využití nabízí otevřené operační systémy jako Symbian nebo Windows Mobile.

Cílem této práce je naprogramovat aplikaci pro mobilní telefon, schopnou změřit vzdálenost, kterou přístroj urazí nad vhodným povrchem. Pro splnění toho cíle je klíčové využití metod pro zpracování digitálního obrazu.

Zpracování digitálního obrazu je velice rozsáhlým tématem, vzhledem k cíli této práce je vhodné zaměřit se pouze na jednu část a to analýzu pohybu.

Analýze pohybu se ve svých publikacích věnuje celá řada autorů zahraničních (např. [7],[1]), ale i tuzemských ([4],[5]). Jedna z metod detekce pohybu úzce souvisí s metodami hledání významných bodů, také na toto téma lze nalézt velké množství prací (např. část [4] nebo práce [9]). V tomto textu jsou představeny základní metody detekce pohybu a popsáno jejich fungování. Znalost těchto metod je zásadní pro správný výběr algoritmu a tedy pro úspěšné splnění cíle práce.

Neméně důležitý je výběr vhodné platformy pro vývoj mobilních aplikací. Platforem existuje několik a každá nabízí různé možnosti, má své výhody i nevýhody. Je třeba zvážit jejich potenciál a poté vybrat takovou, která je nejvhodnější pro naše účely. Společně s podmínkou vhodnosti platformy by měl být splněn požadavek na co nejlepší přenositelnost aplikace. Důležitým kritériem pro výběr paltformy je tedy i její rozšíření.

Práce je rozdělena do čtyř částí.

- V první části si rozebereme možnosti detekce pohybu a měření vzdálenosti pomocí metod zpracování obrazu.
- Druhá část zkoumá možnosti a způsoby implementace aplikací pro mobilní telefony obecně i se zaměřením na zpracování obrazu.
- Třetí část se zabývá návrhem a implementací výsledné aplikace.

- A konečně poslední, čtvrtá, část shrnuje výsledky práce a testování, případně porovnává s jinými podobnými aplikacemi.

Na úvod je třeba definovat, co je v této práci míněno termínem *mobilní telefon*. Tento termín pojmenovává širokou skupinu přístrojů dnes častěji označovaných názvy „chytrý telefon“ (smartphone), komunikátor, ale i úžeji pojatým pojmenováním „mobilní telefon“.

# Kapitola 2

# Zpracování obrazu

Digitální zpracování obrazu obvykle rozdělujeme do několika navazujících částí. Nejdříve je nutné získat obraz reálného světa a tento obraz převést do číslicové podoby.

Snímaní obrazu je v současnosti prováděno pomocí CMOS nebo CCD světločivného snímače. Elektrický náboj vygenerovaný v jednotlivých prvcích snímače je pomocí AD převodníku a mikroprocesoru převeden na jednotlivé obrazové body digitálního snímku(pixely). Rozlišení snímku je úměrné počtu jednotlivých prvků na snímači. Více o obrazových snímačích lze nalézt v [19].

Ve fotoaparátech mobilních telefonů jsou používány snímače typu CMOS, jejich výhody, jako rychlost, nízká cena, nízká spotřeba a jednoduchost výroby, jsou vykoupeny vysokým podílem šumu a nižší obrazovou kvalitou, to ale u mobilních zařízení nemá zásadní váhu.

V digitálních zařízeních je obraz obvykle uložen jako soubor hodnot obrazové funkce pro jednotlivé body obrazu. Obraz může být buď monochromatický, kdy je obrazová funkce, jejíž hodnoty uchováváme, pouze jedna a určuje obvykle intenzitu jasu v každém bodě. V případě barevného obrazu je nutné použít více obrazových funkcí, nejčastěji se používají tři - pro úrovně intenzity jednotlivých barev, které lidské oko vnímá, tedy červenou, zelenou, a modrou. Intenzitu jasu lze pomocí intenzit jednotlivých barev vyjádřit následujícím vztahem:

$$I \quad = \quad 0.299R + 0.587G + 0.114B$$

Hodnoty jsou různé z důvodu rozdílného vnímaní jednotlivých barev lidským okem.

Při digitálním zpracování obrazu se obvykle používá pouze obrazová funkce vyjadřující intenzitu jasu. Je tomu tak proto, že použití jedné funkce je jednodušší než použití tří funkcí a zároveň je jasová funkce plně dostačující pro většinu operací s obrazem.

## 2.1 Metody zpracování obrazu a detekce pohybu

Po digitalizaci obrazu následuje jeho předzpracování. Cílem předzpracování je upravit obraz do podoby vhodné pro následující zpracování, nebo zpracování usnadnit zvýrazněním podstatných rysů. Příkladem předzpracování obrazu může být například filtrace šumu, nebo detekce hran.

Postup po předzpracování se již liší podle účelu, za jakým se obrazem zaobíráme. V případě detekce a měření pohybu v obraze můžeme s úspěchem využít dva základní přístupy. První z nich by se dal laicky popsat tak, že nejprve zjistíme, zda je v obraze pohyblivý

prvek, a teprve potom zjišťujeme, o jaký prvek se jedná. Druhý způsob je přesně opačný, nejprve najdeme objekty v obraze, a potom zjistíme, jestli se pohybují.

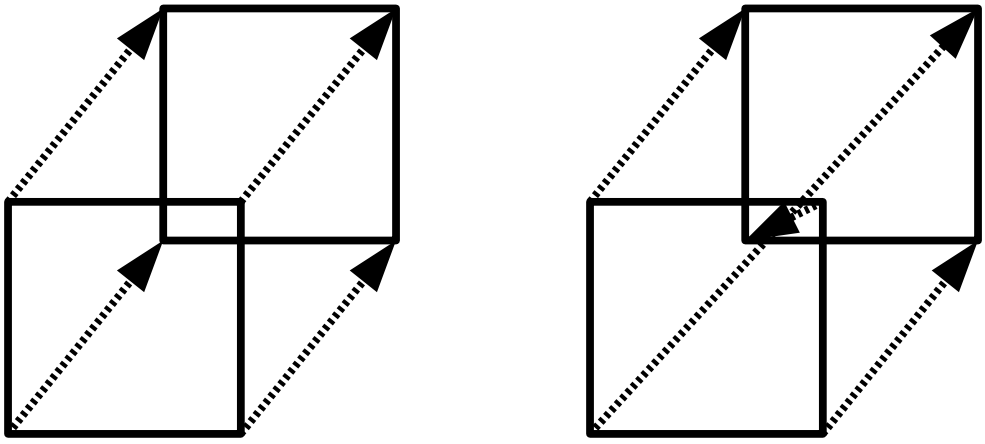
Pro první případ se s úspěchem využívají různé metody odhadu optického toku, což je určení pohybu na základě změny intenzity jasu v jednotlivých bodech obrazu. Druhý případ vychází ze segmentace obrazu například pomocí hledání hran nebo rohů a následného hledání koherence v následujícím snímku. Z uvedeného vyplývá, že pro detekci pohybu je nutné mít alespoň dva po sobě následující snímky.

### 2.1.1 Optický tok

Optický tok je aproximace pohybu v posloupnosti obrazů. Výpočtem optického toku získáme pole vektorů rychlostí, jakou se pohybují jednotlivé body snímku vzhledem ke snímku následujícímu. Optický tok vyjadřuje pohyb v obraze, nikoli nutně ve snímané scéně. Podobnou definci optického toku uvádějí téměř všechny použité publikace.

Základní problémy s určením pohybu ve scéně pomocí optického toku jsou dva[8]. První, z hlediska této práce podružný, se týká toho, že v případě snímání trojrozměrné scény do dvojrozměrného obrazu nejsme schopni rozeznat pohyb bodu ve směru k pozorovateli nebo od pozorovatele. Vzhledem k tomu, že v této práci se budeme věnovat měření pohybu pouze ve dvou rozměrech, nemusí nás tento problém zajímat.

Druhý problém má základ v samotném způsobu určování optického toku. Optický tok vyjadřuje posun jasu (intenzity) v obraze, tento posun lze však sledovat pouze v oblastech, kde je intenzita odlišná od okolí, tedy její gradient je nenulový, avšak není jisté, zda se pohybuje pouze ona oblast, nebo zda se pohybuje celá scéna. Tento problém se nazývá *problém apertury*. Pokud se přesněji podíváme, co je to optický tok, zjistíme, že ho můžeme považovat za vektorové pole[8], které definuje zobrazení všech bodů z obrazu $A$ do obrazu $B$, přičemž předpokladem je zachování intenzity (viz dále). Takových zobrazení však existuje nekonečné množství. Příkladem problému apertury je obrázek 2.1.



Obrázek 2.1: Problém apertury - různé správné možnosti pohybu v obraze[8]

### Určování optického toku

Všechny metody pro výpočet optického toku vycházejí z předpokladu zachování intenzity (jasu). Mějme funkci $I(x, y, t)$, která představuje intenzitu $I$ v bodě $(x, y)$ v čase $t$, potom předpoklad zachování intenzity můžeme vyjádřit vztahem

$$I(x, y, t) \quad = \quad I(x + u\delta t, y + v\delta t, t + \delta t) \tag{2.1}$$

kde vektor $(u, v)$ je vektor optického toku pro daný bod.

Metody pro výpočet optického toku můžeme rozdělit do několika skupin. Toto dělení uvádí například Barron [1].

- Diferenční metody
- Vyhledávání oblastí
- Přístupy založené na energii
- Metody založené na fázi

Blíže jsou tyto přístupy charakterizovány například v [1], nebo [8], v této práci se budu věnovat pouze metodě Horna a Schuncka popsané v [7] a její úpravě [6].

### Metoda Horna a Schuncka

Tato metoda je, jak již bylo zmíněno výše, popsána v dokumentu *Determining Optical Flow*[7]. Její popis v této práci z uvedeného dokumentu vychází.

Metoda Horna a Schuncka stejně jako jiné, vychází z předpokladu zachování intezity, tedy položme

$$\frac{\mathrm{d}I(x, y, t)}{\mathrm{d}t} \quad = \quad 0 \tag{2.2}$$

což lze pomocí parciálních derivací rozepsat jako

$$I_x \frac{\mathrm{d}x}{\mathrm{d}t} + I_y \frac{\mathrm{d}y}{\mathrm{d}t} + I_t \quad = \quad 0 \tag{2.3}$$

Pokud si dále dosadíme

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u \quad a \quad \frac{\mathrm{d}y}{\mathrm{d}t} = v$$

dostaneme první rovnici pro určení optického toku

$$I_x u + I_y v + I_t \quad = \quad 0 \tag{2.4}$$

kde neznámé $(u, v)$ jsou komponenty vektoru rychlosti toku a $I_x, I_y, I_t$ parciální derivace intenzity obrazu podle $x, y$ a $t$.

Pro určení dvou neznámých není však jedna rovnice dostačující, proto Horn a Schunck zavádějí podmínku plynulé změny optického toku[7]. Podmínka vychází z představy, že části jednotlivých objektů se pohybují v závislosti na sobě, tedy že objekt se pohybuje jako celek a proto je optický tok v bodech, které náleží jednomu objektu podobný nebo stejný.

Plynulost změny je v metodě Horna a Schuncka vyjádřena Laplaciány komponent $x$ a $y$ optického toku $(u, v)$.

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{2.5}$$

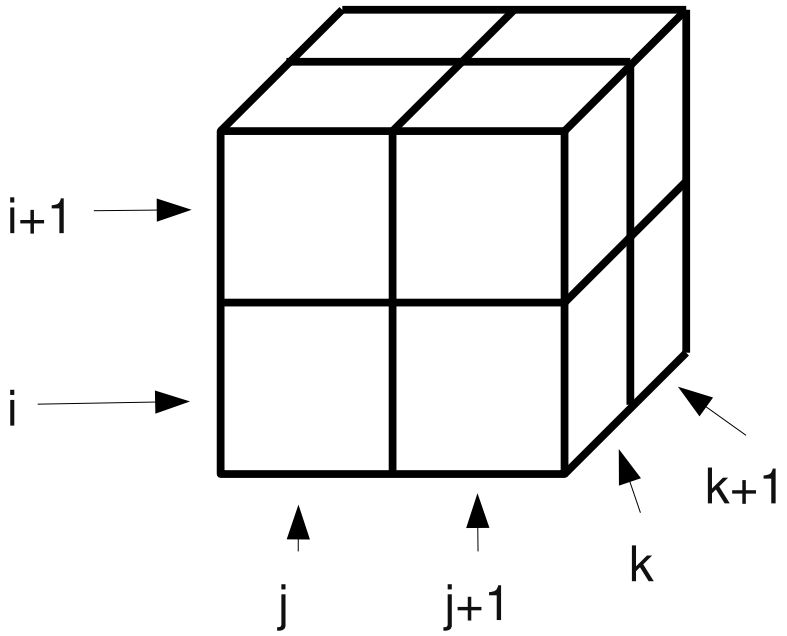
$$\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{2.6}$$

V ideální situaci naprosto plynulé změny optického toku jsou oba Laplaciány nulové, proto je cílem metody najít takové $u, v$, při kterém budou oba Laplaciány co nejmenší. Horn a Schunck toho dosahují minimalizací čtverce magnitudy gradientu rychlosti optického toku.

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \tag{2.7}$$

$$\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.8}$$

Nyní tedy známe teoretický základ a můžeme přejít k numerickému řešení. Nejprve je třeba určit parciální derivace intenzity $I_x, I_y, I_t$. Horn a Schunck považují za nejvhodnější metodu, která aproximuje výsledek pro střed pomyslné krychle vytvořené ze čtyř pixelů jednoho a čtyř pixelů následujícího snímku. Pixely jsou znázorněny na obrázku 2.2.




Obrázek 2.2: Sloupce $j$ představují osu $x$ jednotlivých snímků, řady $i$ osu $y$ a $k$ představuje časovou osu, tedy jednotlivé snímky[7].

$$
\begin{aligned}
I_x &\approx \frac{1}{4}\{I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} \\
&\quad + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\} \qquad (2.9)\\
I_y &\approx \frac{1}{4}\{I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} \\
&\quad + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\} \qquad (2.10)\\
I_x &\approx \frac{1}{4}\{I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} \\
&\quad + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\} \qquad (2.11)
\end{aligned}
$$

Dále potřebujeme odhadnout hodnoty Laplaciánů $u$ a $v$

$$
\begin{aligned}
\nabla^2 u &= \ni(\bar{u}_{i,j,k} - u_{i,j,k}) \qquad (2.12)\\
\nabla^2 v &= \ni(\bar{v}_{i,j,k} - v_{i,j,k}) \qquad (2.13)
\end{aligned}
$$

kde $\bar{u}$ a $\bar{v}$ jsou vážené průměry okolních hodnot dané konvolučním jádrem:

$$
\begin{pmatrix}
\frac{1}{12} & \frac{1}{6} & \frac{1}{12} \\
\frac{1}{6} & 0 & \frac{1}{6} \\
\frac{1}{12} & \frac{1}{6} & \frac{1}{12}
\end{pmatrix}
$$

Hodnoty $\bar{u}$ a $\bar{v}$ je možno určit následujícím způsobem:

$$
\begin{aligned}
\bar{u}_{i,j,k} &= \frac{1}{6}\Big\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k})\Big\} \\
&= \frac{1}{12}\Big\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k})\Big\} \qquad (2.14)\\
\bar{v}_{i,j,k} &= \frac{1}{6}\Big\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k})\Big\} \\
&= \frac{1}{12}\Big\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k})\Big\} \qquad (2.15)\\
& \qquad (2.16)
\end{aligned}
$$

Nyní si můžeme jednotlivé rovnice představit jako chybové členy. Chybu pro změnu intenzity v obraze můžeme vyjádřit jako

$$
\rho_b = I_x u + I_y v + I_t \qquad (2.17)
$$

a chybový člen pro plynulost změny toku vyjádříme takto:

$$
\rho_b^2 = \Big(\frac{\partial u}{\partial x}\Big)^2 + \Big(\frac{\partial u}{\partial y}\Big)^2 + \Big(\frac{\partial v}{\partial x}\Big)^2 + \Big(\frac{\partial v}{\partial y}\Big)^2 \qquad (2.18)
$$

Naším cílem je minimalizovat celkovou chybu vyjádřenou součtem obou předchozích chybových členů[7].

$$
\rho^2 = \alpha^2 \rho_c^2 + \rho_b^2 \qquad (2.19)
$$

Koeficient $\alpha^2$ představuje váhu ve prospěch homogenity optického toku v oblastech s nízkou změnou intenzity. Tím pomáhá distribuovat dobře odhadnutý optický tok do míst se špatnými podmínkami pro jeho určování.

Pro minimalizaci celkové chyby nejprve určíme parciální derivace $\rho$ podle jednotlivých složek $u$ a $v$. S použitím aproximace Laplaciánů uvedené výše získáme soustavu dvou rovnic o dvou neznámých, jejíž řešením je :

$$u = \frac{(\alpha^2 + I_y^2)\bar{u} - I_x I_y \bar{v} - I_x I_t}{\alpha^2 + I_x^2 + I_y^2} \tag{2.20}$$

$$v = \frac{(\alpha^2 + I_x^2)\bar{v} - I_x I_y \bar{u} - I_y I_t}{\alpha^2 + I_x^2 + I_y^2} \tag{2.21}$$

Nyní tedy máme soustavu rovnic pro výpočet optického toku v jednom bodě obrazu, ale protože hodnota toku v jednom bodě závisí i na hodnotě optického toku v okolních bodech, je nutné počítat optický tok pro všechny body najednou. Z toho důvodu musíme sestavit rovnice pro výpočet $u_{0\ldots n}, v_{0\ldots n}$, kde $n$ je počet bodů v obraze. Pro řešení takovéto rozsáhlé soustavy rovnic Horn a Schunck doporučují použít jednu z iterativních metod, například Gauss-Seidelovu. Pro výpočet nových hodnot $u_{k+1}, v_{k+1}$ v každé iteraci používají následující vztahy[7]:

$$u_{k+1} = \frac{\bar{u}_k - I_x[I_x \bar{u}_n + I_y \bar{v}_n I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.22}$$

$$v_{k+1} = \frac{\bar{v}_k - I_y[I_y \bar{u}_n + I_y \bar{v}_n I_t]}{\alpha^2 + I_x^2 + I_y^2} \tag{2.23}$$

Posledním problémem zůstává určit, kdy zastavit iteraci. Jednou z možností je pokles relativní chyby pod danou mez. Tento způsob s sebou však nese náklady na výpočet chybového členu v každém kroku. Jinou možností je počet iterací závislý na velikosti oblasti se špatně odhadnutelným optickým tokem. Důležité je nepodceňovat určení počátečních hodnot iterace. Tyto lze odhadnout například z předchozích snímků - lze předpokládat, že pokud na předchozích snímcích optický tok mířil jedním směrem, na následujícím snímku bude mířit podobně. Vhodným odhadem lze snadno a přitom velice razantně snížit počet iterací[7].

### Konstatní optický tok

Předchozí metoda je nejvhodnější pokud chceme měřit a zjišťovat informace o pohybu jednotlivých objektů v obraze, jejím výsledkem je totiž hodnota optického toku pro každý jednotlivý pixel. Avšak pro případy, kdy můžeme předpokládat, že celá scéna v obraze se pohybuje stejně, existuje výpočetně méně náročná metoda. Její autorem je Berthold K.P. Horn a je popsána v dokumentu *Determining Constant Optical Flow*[6].

Vychází z podobných základů jako výše popsaný výpočet. Předpokladem je opět zachování intenzity obrazu v čase vyjádřené rovnicí 2.4. V případě, že hodnota optického toku je v každém bodě obrazu stejná, je tato rovnice dostačující pro její určení.

Pro odhad můžeme použít pouze rovnice ze dvou bodů obrazu, avšak spolehlivost takového výsledku je při použití na reálných zarušených obrazech limitně blízká nule, proto Horn doporučuje použít rovnice ze všech obrazových bodů ve snímku. Toto můžeme vyjádřit sumou rovnice intenzity ve všech bodech obrazu[6]:

$$\sum_{i=1}^{n-1} \sum_{j=1}^{m-1} \Big( u I_x + v I_y + I_t \Big)^2 \mathrm{d}x\mathrm{d}y \tag{2.24}$$

kde $m$ je počet pixelů ve sloupci, $n$ počet pixelů na řádku a $I_x, I_y, I_t$ jsou parciální derivace podle jednotlivých složek, jejichž aproximace je popsána v rovnicích 2.9 – 2.11. V ideálním případě bez jakýchkoli chyb měření by tato suma byla nulová, toho však nelze reálně dosáhnout, proto se budeme snažit sumu alespoň minimalizovat. Výsledkem minimalizace je soustava rovnic s výsledkem:

$$u = \frac{\sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x I_y \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_y I_t - \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_y^2 \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x I_t}{\nabla} \tag{2.25}$$

$$v = \frac{\sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x I_y \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x I_t - \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x^2 \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_t I_t}{\nabla} \tag{2.26}$$

přičemž determinant soustavy je vyjádřen:

$$\nabla = \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x^2 \sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_y^2 - \left(\sum_{i=1}^{n-1}\sum_{j=1}^{m-1} I_x I_y\right)^2 \tag{2.27}$$

Tato soustava nemá řešení pouze v případě, že determinant je roven nule, což nastane tehdy, když gradient intenzity je stejný ve všech bodech obrazu[6].

Jak je z popisu vidět, tato metoda je vzhledem k nižší výpočetní náročnosti vhodnější, než výše uvedený komplexní přístup, ovšem pouze pro specifické účely. Podobný výpočet se s úspěchem používá například v optických myších.

### 2.1.2 Detekce pohybu s využitím rozpoznávání významných bodů

Tuto metodu detailně popsují Šonka, Hlaváč a Boyle v *Image processing, Analysis and Machine Vision*[5] Hlavním problémem při použití této metody je zjištění vzájemné korespondence jednotlivých objektů a jejich částí v obraze. Nejprve je však třeba nalézt významné body, podle kterých lze rozpoznat jednotlivé objekty. Existuje velké množství algoritmů pro hledání významných bodů, které dávají velmi rozdílné výsledky. Některé jsou velice přesné a robustní, jiné naopak velmi rychlé.

#### Metody rozpoznávání významných bodů

Pro účely detekce významných bodů pro rozpoznávání pohybu je jedním z nejvhodnějších algoritmů tzv. Moravcův operátor.

$$moravec(i,j) = \frac{1}{8}\sum_{k=i-1}^{i+1}\sum_{k=i-1}^{i+1} |f(i,j) - f(k,l)| \tag{2.28}$$

Tato metoda sice mezi vrcholovými detektory nepatří k nejlepším, protože často za vrchol označuje i hranu, je citlivá na šum a není rotačně invariantní, ale pro naše účely se hodí, je totiž výpočetně nenáročná, navíc byla s podobným záměrem původně navržena. Použití Moravcova operátoru pro účely detekce pohybu doporučují i Hlaváč a Šonka v knize *Počítačové vidění*[4]. Lze samozřejmě použít i jiné rohové nebo hranové detektory (přehled těch hlavních lze nalézt v Kaněčkově práci[9] nebo na Wikipedii[18]), avšak pro analýzu pohybu v reálném čase je klíčová rychlost výpočtu, proto je nutné při výběru detektoru brát v úvahu toto hledisko.

**Určení vzájemné korespondence**

Určování vzájemné korespondence je řízeno třemi principy.

1. Rozlišitelnost jednotlivých bodů od sebe
2. Podobnost vzájemně korespondujících bodů
3. Konzistence jednotlivých korespondencí

Se znalostí těchto principů můžeme nastínit algoritmus (pro detailnější popis viz [5]):

- Nejprve pomocí detektoru nalezneme významné body.
- Pro každý bod z prvního snímku spočítáme pravděpodobnost jeho shody s každým z bodů druhého snímku do určené vzdálenosti.
- Pro každý bod vybereme nejlepší shodu.
- Nakonec se podíváme, jestli shody nekolidují s principem konzistence, tedy jestli se náhodou dva body jednoho objektu nepohybují odlišně.

Použití této metody je vhodné pro hledání pohybujících se objektů v obraze, její výpočetní náročnost však kvůli opakovanému procházení obrazem nepatří ani s použitím rychlého detektoru k nejnižším.

# Kapitola 3

# Programování aplikací pro mobilní telefony

Výkon mobilních telefonů během posledních let rostl neuvěřitelným tempem a spolu s ním vzrůstal i segment programování mobilních aplikací. Nejprve se ale podívejme do historie. V polovině devadesátých let s rozvojem sítě GSM se začaly objevovat mobilní telefony, které měli více funkcí, než je volání nebo SMS. Objevily se první budíky, kalendáře nebo hry. Zpočátku byly veškeré aplikace vestavěné ve firmwaru přístroje, postupně však výrobci začali přicházet s různými způsoby doinstalování aplikací. Velký rozmach tohoto odvětví začal kolem roku 2000, kdy se v přístrojích poprvé objevily implementace platformy Java od společnosti Sun Microsystems, v té době již dobře známě na osobních počítačích.

Java poskytuje široké spektrum možností, ale má i spoustu omezení. Obzvláště její první implementace pro mobilní zařízení Java Micro Edition byla oproti Javě pro osobní počítače silně limitovaná. Proto již v počátcích přišli výrobci s jinými platformami, které byly určené převážně pro hry, například Mophun nebo InFusio. Tyto platformy, ač pro svůj účel jistě vhodnější než Java, však nestačily jejímu překotnému vývoji a nakonec pro nedostatečnou podporu mezi výrobci nebo z důvodu malého množství vývojářů zanikly a přenechaly pole působení společnosti Sun. V současnosti lze říci, že téměř každý mobilní telefon podporuje platformu Java.

Téměř od počátku však existoval i jiný směr vývoje mobilních aplikací, než dnes představuje Java. Přiblížit mobilní telefon běžnému počítači se pokoušeli vývojáři různých otevřených operačních systémů. Otevřený v tomto případě znamená, že umožňuje volně doinstalovat další aplikace. V současnosti je největším hráčem na trhu Symbian OS, instalovaný do přístrojů společností Nokia, Sony-Ericsson nebo Samsung. Velký rozmach poslední dobou zaznamenal operační systém Microsoft Windows Mobile, který do svých telefonů instaluje hlavně společnost HTC, ale i mnoho dalších menších výrobců. Existují i další operační systémy obvykle na bázi Linuxu, jejichž vývoji se věnuje například Motorola.

Menší revoluci v poslední době způsobila firma Apple se svým telefonem iPhone. Operační systém tohoto přístroje však není plně otevřen vývojářům aplikací, vývojový kit je sice k dispozici zdarma, ale aplikace mohou využívat jen omezené zdroje a celkově musí splňovat velmi tvrdé licenční podmínky.

Ve výhledu do budoucna lze předpokládat pravděpodobný velký rozvoj operačního systému Android sdružení Open Headset Alliance, mezi jehož členy patří například Google, HTC, Motorola nebo Qualcomm. Tento systém založený na Linuxu v sobě spojuje komplexnost Symbianu a Windows Mobile s filozofií Javy a navíc přidává skutečnou otevřenost

na úrovni zdrojových kódu, nebo možnost nahradit jakoukoli, i systémovou, aplikaci.

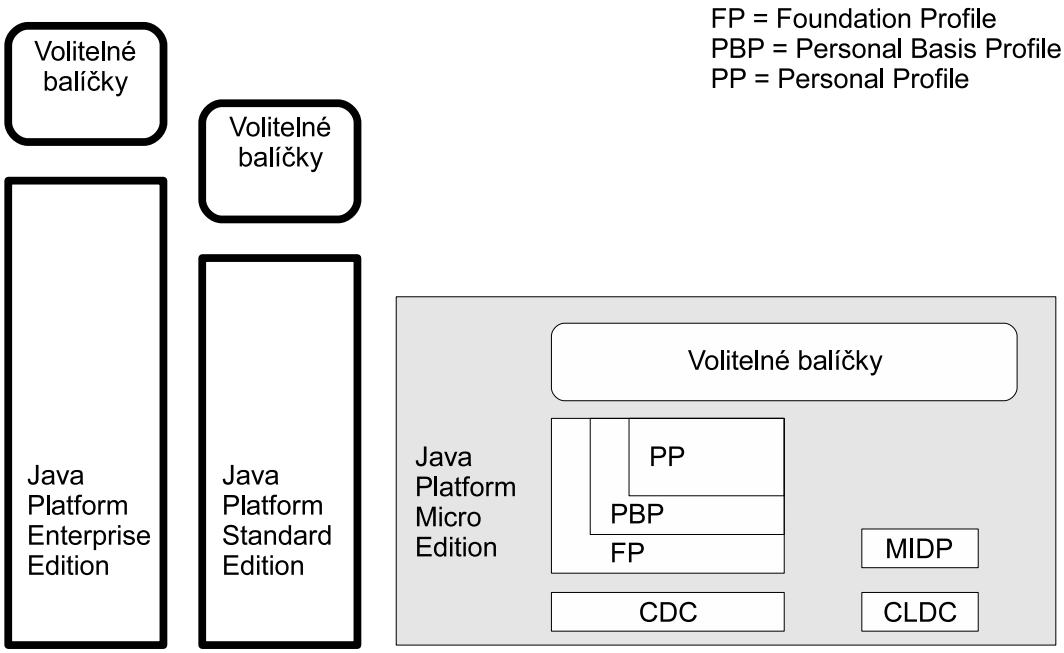
Nyní se blíže podíváme na nejrozšířenější platformy, jejich filozofii, možnosti a omezení a podrobněji také na jejich potenciál v oblasti využití multimédií.

## 3.1 Java Micro Edition

Java ME (také JME) původně vznikla jako platforma odvozená od standardní Javy určená pro malá přenosná zařízení. Je vyvíjena společností Sun Microsystems zhruba od roku 2000 jako náhrada za PersonalJava, která podobným účelům sloužila předtím.

Java ME existuje ve dvou základních variantách, které se liší použitím. První z nich je *Connected Device Configuration (CDC)*, které je určené pro výkonnější zařízení s důrazem na kompatibilitu se standardní Javou. CDC podporuje kompletní virtuální stroj Java a všechny základní knihovny Java SE, pouze některé z nich jsou upraveny pro použití na méně výkonných zařízeních, než běžné počítače. Celkově CDC funguje na zařízeních, která mají alespoň 2 MB operační paměti pro virtuální stroj Java a 2 MB ROM.

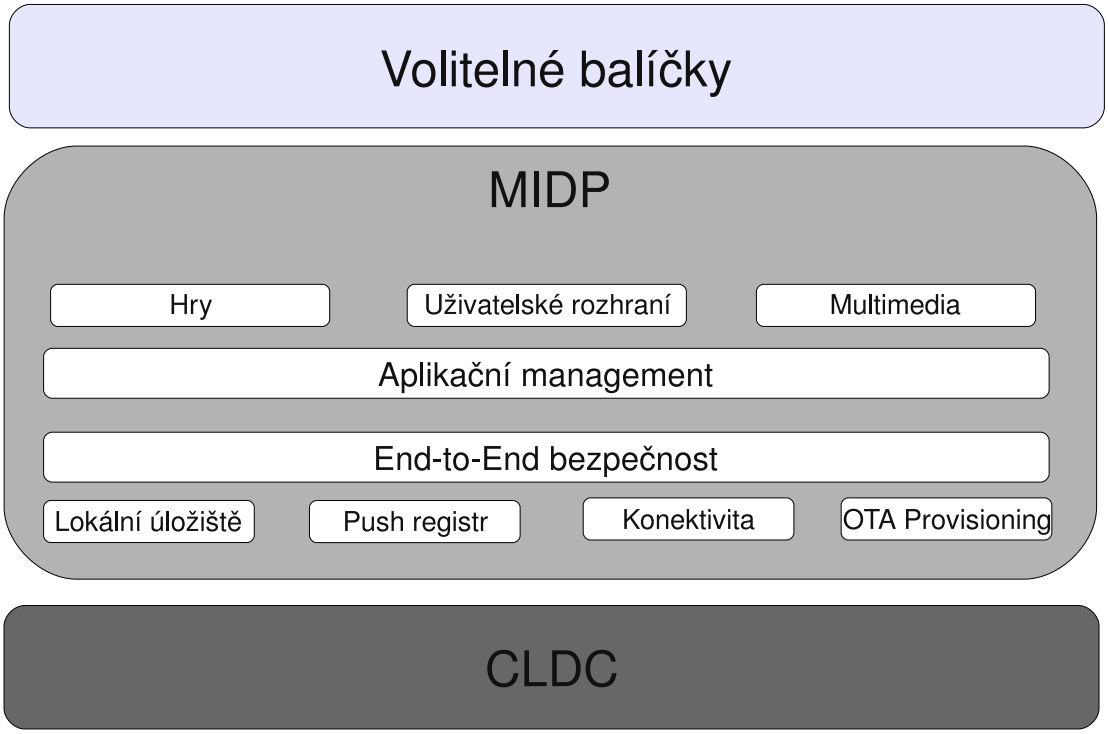
Varianta pro ještě více omezené přístroje je známá pod názvem *Connected Device Limited Configuration (CLDC)*. Původní specifikace CLDC byla určena pro zařízení s 128 až 256 KB RAM, jakou disponovaly například tehdejší low-end mobilní telefony. Některé vlastnosti standardní Javy zde úplně chyběly, například podpora čísel s plovoucí řádovou čárkou. V současnosti s rostoucím výkonem mobilních zařízeních se podpora většiny chybějících součástí dostává i do specifikace CLDC. Vzhledem k tomu, že na mobilních telefonech se vyskytuje téměř výhradně specifikace CLDC, budu se v dalším textu věnovat jen jí.




Obrázek 3.1: Architektura platformy Java[15]

Součástí implementace JME v mobilních telefonech je kromě spíše nízkoúrovňového

CLDC i balík vysokoúrovňových API známý pod názvem *Mobile Information Device Profile (MIDP)*. MIDP poskytuje vývojářům API pro přístup k datovým úložištím, konektivitě, bezpečnosti, multimédiím nebo uživatelskému rozhraní. Strukturu MIDP ukazuje obrázek 3.2 Různí výrobci samozřejmě implementují Javu různě, ale všichni dodržují specifikace




Obrázek 3.2: Podsystémy a služby MIDP[16]

CLDC a MIDP a změny jsou představovány pouze přidáním vlastních nebo volitelných balíků či API. Například přístroje společnosti Nokia kromě standardních CLDC a MIDP nabízejí i API pro 3D grafiku, ovládání pohybových senzorů nebo SIP a také vlastní Nokia UI API a Nokia SMS API.

Propracovanou podporu multimédií, hlavně zvuků, videa a integrovaného fotoaparátu poskytuje volitelný balík *Mobile Media API (MMAPI)*, který je však dnes součásti téměř každé implementace JME na mobilních telefonech. MMAPI nabízí podporu přehrávání i záznamu rozličných zvukových formátů, přehrávání i záznam videa pomocí integrované kamery, nebo snímání jednotlivých snímků.

Snímání jednotlivých snímků bývá však velice pomalé a video lze zaznamenávat pouze do určitého cíle, například souboru a nelze ho během záznamu zpracovávat. Tyto vlastnosti zásadně snižují použitelnost Javy ME pro vývoj aplikací zpracovávajících obraz, či spíše obrazové sekvence.

Pro vývoj jiných aplikací je však Java ME výborně použitelná, protože přináší objektovou orientaci, snadný návrh a implementaci, stabilní běh a přenositelnost na úrovni Byte kódu (za předpokladu existence všech potřebných balíku na cílovém zařízení), ostatně jako implementace Javy pro osobní počítače. Jedinou dílčí nevýhodou je určitá pomalost aplikací podmíněná nutností běhu ve virtuální stroji.

Co se týče vývojových nástrojů existuje široká škála integrovaných prostředí, ať už

společností Sun podporované NetBeans nebo Eclipse, a další. Společnost Sun také poskytuje základní emulátor, který dokáže napodobit veškeré API implmentované v CLDC a MIDP a další přídavné balíky společnosti SUN, jako například MMAPI. Jednotlivé společnosti vyrábějící mobilní telefony dále nabízejí emulátory či celé SDK podporující specifika jejich vlastní implementace JME.

## 3.2 Otevřené operační systémy

### 3.2.1 Symbian

Symbian vychází z operačního systému EPOC firmy PSION, který se v původní verzi objevil roku 1989. Z počátku šlo o 16 bitový operační systém pro přenosné organizéry značky PSION, v polovině devadesátých let se však objevila 32 bitové verze EPOC32, která se později stala základem Symbian OS. Společnost Symbian vznikla roku 1998 ze softwarové divize společnosti PSION za podpory hlavních hráčů na trhu mobilních telefonů - Nokia a Ericsson.

Jak již bylo zmíněno výše, operační systém Symbian vychází z EPOC32 a navazuje i na číslování jeho verzí, první verzí Symbianu byla tedy verze 6 vydaná v roce 2001. Již v této verzi se objevila dvě různá uživatelská rozhraní, jedno určené pro přístroje s dotykovým displejem, ze kterého se později vyvinulo rozhraní QUI používané společností hlavně Sony Erricsson a Nokia S80, a druhé pro přístroje pouze s klávesnicí, ze kterého vychází dnešní Nokia S60. Postupně vznikaly novější a novější verze Symbianu[17], v současnosti se v telefonech objevují verze 9.2 (Nokia S60 3rd Edition Feature Pack 1) a 9.1 (Sony Ericsson QUI3), přičemž poslední vydanou verzí Symbianu je 9.5.

Programovaní aplikací pro Symbian OS je možné v několika programovacích jazycích. V prvé řadě je možné programovat nativní aplikace v jazyce C++, Symbian také samozřejmě obsahuje podporu Javy ME, platforma S60 také po instalaci přídavného balíčku podporuje Python. Implementace Javy se nijak neliší od Javy ME v běžných mobilních telefonech. Python pro S60[13] obsahuje základní API pro běžné aplikace, například GUI, posílaní SMS, přístup do kontaktů a kalendáře, Bluetooth, audio a také zjednodušené API pro ovládání integrované kamery.

Použití jazyka C++ zaručuje úplný přístup ke všemu, co mobilní telefon se Symbianem nabízí. Je možno ovládat jakékoli integrované hardwarové zařízení přístroje, umožňuje i komplexní přístup ke všem softwarovým prostředkům, na všechno existují velice propracovaná API a knihovny. Cenou za tyto obrovské možnosti je však složitost vývoje aplikací v C++.

Celý systém Symbian je napsán v jazyce C++, není to však stejné C++, jaké známe z osobních počítačů. Při vývoji aplikací pro mobilní telefony, je třeba brát v úvahu, že podobné aplikace mohou běžet velice dlouhou dobu bez vypnutí a restartu systému, proto musíme klást velký důraz na efektivní práci s pamětí. Důležité je také poctivě zpracovat všechny chyby, které mohou nastat. Pro Symbian je také, narozdíl od běžného C++, velice podstatný počet bitů základních datových typů, naopak nevyužije podporu plnohodnotné vícenásobné dědičnosti v jazyce C++.

Problémy s úniky paměti jsou v systému Symbian řešeny kladením důrazu na práci s objekty, jejich vytváření a rušení. Únik paměti může také nastat při nedostatku paměti pro vytvoření nového objektu, toto je v Symbianu řešeno dvoufázovou konstrukcí, kdy v první fázi alokujeme samotnou třídu a teprve ve druhé fázi objekty, které vlastní. Kdybychom totiž alokovali všechny objekty hned v konstruktoru, kdy ještě nemáme k dispozici ukazatel

na naši třídu a došlo by k problému s pamětí, neměli bychom příležitost naší alokovanou třídu uvolnit.

Vzhledem k tomu, že základy Symbianu byly položeny v době, kdy v C++ ještě neexistovalo, nebo nefungovalo příliš dobře zachytávání výjimek, má Symbian vyvinut vlastní systém, jak si s podobnými chybami poradit, jde o tzv. *generování leave* a jeho zachytávaní pomocí makra `TRAP`, které je podobné bloku `try catch` v dnešním C++ nebo Javě. Souvislost s *leave* má i použití tzv. úklidového zásobníku *CleanupStack*.

Blíže se programovaní aplikací pro Symbian věnuje například kniha *Programujeme aplikace pro Symbian OS v jazyce C++*[3], existuje samozřejmě i jiná literatura, například společnost Nokia má na svých stránkách velice kvalitně zpracované veškeré informace pro vývojáře (viz [11]).

Základem multimediálních služeb Symbianu je Multimédia Framework (MMF), jednotný systém zpracování a použití veškerých multimédií. Architektura tohoto systému vychází z vrstvy vysokoúrovňových klientských API navazujících na soustavu kontrolérů, kterými jsou ovládána jednotlivá nízkoúrovňová rozhraní a hardwarové zdroje.

Programování multimediálních aplikací vychází z návrhového vzoru *observer*[3]. Ten používá třídu, která kromě vlastních metod určitého API definuje i virtuální metody poskytující zpětnou vazbu směrem k programátorovi. Pro lepší pochopení si popišme případ použití integrované kamery. Pro vyfocení jednotlivého snímku má observer kamery metodu `Snap()`, pokud tuto metodu zavoláme kamera vytvoří snímek a poté zavolá virtuální metodu observeru `ImageReady()`, ve které musí programátor sám definovat, co se má s vytvořeným snímkem provést.

Pro vývoj aplikací pro Symbian je přímo určeno několik vývojových prostředí, mezi nejpoužívanější z nich patří Metrowerks CodeWarrior a Nokia Carbide.c++, s plug-inem Nokia Carbide.vs, lze použít i Microsoft Visual Studio 2005. Vývojové prostředí Carbide vychází z oblíbeného Eclipse a je určeno převážně pro vývoj S60 aplikací. Dále existují různé verze SDK s emulátory. Například společnost Nokia pravidelně vydává aktualizované SDK pro aktuální verzi Symbianu S60, také Sony-Ericsson vydává SDK pro QUI.

### 3.2.2 Windows Mobile

Windows Mobile je operační systém pro mobilní telefony a PDA založený na platformě Windows CE[20], což je varianta Microsoft Windows určená pro malá nebo vestavěná zařízení. Historie Windows CE se začala psát koncem roku 1996, kdy se začaly objevovat první přístroje třídy Handheld, zpočátku tedy nešlo o mobilní telefony. Dnešnímu použití se tehdy ještě Windows CE přiblížila v roce 2000 spolu s představením platformy PocketPC. V roce 2002 se od platformy Windows CE oddělila větev určená pouze pro zařízení PocketPC/Smartphone, byla pojmenována Windows Mobile 2003. O tři roky později následovala Windows Mobile 5.0, v současnosti je aktuální verze 6.0 (6.1)[21]. Existují dva základní druhy Windows Mobile – Smartphone pro zařízení s běžnou klávesnicí a PocketPC pro přístroje s dotykovým displejem.

Programování aplikací pro Windows Mobile je velice podobné vývoji pro Windows na PC, například použití Win32API je na obou platformách takřka shodné, lze samozřejmě využít i .NET Framework, byť v úpravě Compact, jeho funkce je však standardnímu .NET velice podobná. .NET aplikace lze samozřejmě programovat v jazyce C# nebo Visual Basic, čisté Win32 aplikace v C++/Visual C++. Odlišnosti mezi programováním pro Windows Mobile a běžné Windows samozřejmě existují, ale týkají se spíše různých specialit mobilních telefonů, než nějakých rozdílů ve filozofii platforem. Pododrobně se programovaní pro

Windows Mobile věnuje kniha Luboslava Lacka[10], je ale spíše zaměřena na problematiku využití síťových technologií a webových služeb.

Pro vývoj multimediálních aplikací v prostředí Windows Mobile lze s úspěchem využít jak většinu multimediálních funkcí Win32API, tak mobilní verzi DirectX, a dále různé integrované kodeky a rozhraní. Také integrovanou kameru lze ovládat pomocí jednoduchých funkcí.

Hlavním a nejdůležitějším vývojovým nástrojem pro Windows Mobile je Microsoft Visual Studio a dále SDK pro příslušnou verzi WM.

### 3.2.3 Další operační systémy

Vzhledem k tomu, že rozšíření dalších operačních systémů je v našich oblastech mizivé, a některé z nich dokonce ještě nejsou implementovány na žádném prodávaném zařízení, nemá smysl zde rozebírat možnosti, jak pro ně programovat aplikace.

# Kapitola 4

# Návrh a implementace

V této kapitole nejprve na základně analýzy zvolím nejvhodnější řešení daného problému a poté na základě návrhu popíšu implementaci konečné aplikace. Cílem je aplikace fungující na co nejširším spektru mobilních telefonů, proto je velice důležitý výběr implementační platformy, který je v této práci také rozebrán.

## 4.1 Návrh

Kvalita návrhu aplikace je pro vývoj výchozím bodem, od kterého se odvíjí veškeré vlastnostni konečného programu, je proto nutné věnovat návrhu patřičnou pozornost. Lze s úspěchem využít různých návrhových vzorů, které usnadňují vývoj a zároveň zvyšují znovupoužitelnost aplikace.

### 4.1.1 Analýza problému

Aby bylo možné navrhnout co nejlepší řešení daného problému, je nejprve nutné ho důkladně analyzovat. Analýza by měla odpovědět na několik základních otázek:

- Jak zjistíme pohyb v posloupnosti digitálních obrazů?
- Jaký pohyb budeme hledat?
- Jakým způsobem tento pohyb změříme?

Odpovědět na první otázku je vcelku snadné, můžeme říci, že se jedná o pohyb, pokud se intenzita jednotlivých bodů v obraze nějakým způsobem mění nezávisle na ostatních ale zároveň podobně jako okolní body. Je samozřejmě nutné počítat s šumem, jeho vliv je však omezen podmínkou podobnosti změn s okolními body.

Druhá otázka je už složitější. V obrazu se může buď pohybovat celá scéna, nebo pouze jeden objekt (či více objektů). Pokud máme měřit vzdálenost, kterou mobilní telefon urazí nad určitým povrchem, je pravděpodobné, že scéna bude v klidu a pohybovat se bude pozorovatel. Z pohledu pozorovatele to znamená, že pohybovat se bude celá scéna, či veškeré objekty na ní.

Odpověď na třetí otázku můžeme najít v druhé kapitole této práce, která popisuje různé metody analýzy pohybu. Můžeme použít metodu, který využívá detekci významných bodů a předpokládat, že všechny objekty v obraze se pohybují stejným způsobem. Nebo můžeme jednou z metod určit optický tok pro každý bod obrazu a za vektor rychlosti, kterou

se pohybuje celá scéna označit ten, který je nejčastější. Nejvhodnější se však jeví použití metody, která je pro podobné účely přímo určena. Jde o metodu pro určování konstantního optického toku (zde 2.1.1).

### 4.1.2 Výběr vhodné implementační platformy

Základním požadavkem je co nejširší rozšíření platformy, aby byla zaručena co nejlepší přenositelnost. Z tohoto pohledu se nejvýhodněji jeví platforma Java Micro Edition. Problémem Javy ME je však nevhodnost jejího použití pro aplikace zpracovávající obrazové sekvence z integrované kamery. Pro naše účely by bylo nejvhodnější zpracovávat vždy dva po sobě jdoucí snímky, přičemž by mezi nimi měl být co nejmenší časový rozdíl. Jak už bylo zmíněno výše jedinou možností jak získat po sobě následující snímky v Javě ME je použití metody MMAPI pro vytvoření statického snímku, tato metoda je však vcelku pomalá a není schopná dodávat snímky v patřičné frekvenci. Metoda, která by umožnila zpracovávat jednotlivé snímky při natáčení videosekvence v Javě ME neexistuje.

Naše požadavky splňuje modul ECam operačního systému Symbian. Vzhledem k tomu, že Symbian, konkrétně s uživatelským rozhraním S60, je přinejmenším v Evropě druhou nejrozšířenější (hned po Javě) platformou pro programování mobilních aplikací, rozhodl jsem se jeho možností využít.

### 4.1.3 Návrh aplikace

Základním prvkem aplikace bude třída, která bude implementovat algoritmus pro měření vzdálenosti. Ten bude založen na algoritmu pro výpočet konstantního optického toku popsaném v druhé kapitole této práce (část 2.1.1). Jinak bude implementace dodržovat obvyklé postupy používané při vývoji aplikací pro operační systém Symbian. Strukuru aplikace ukazuje obrázek 4.1.

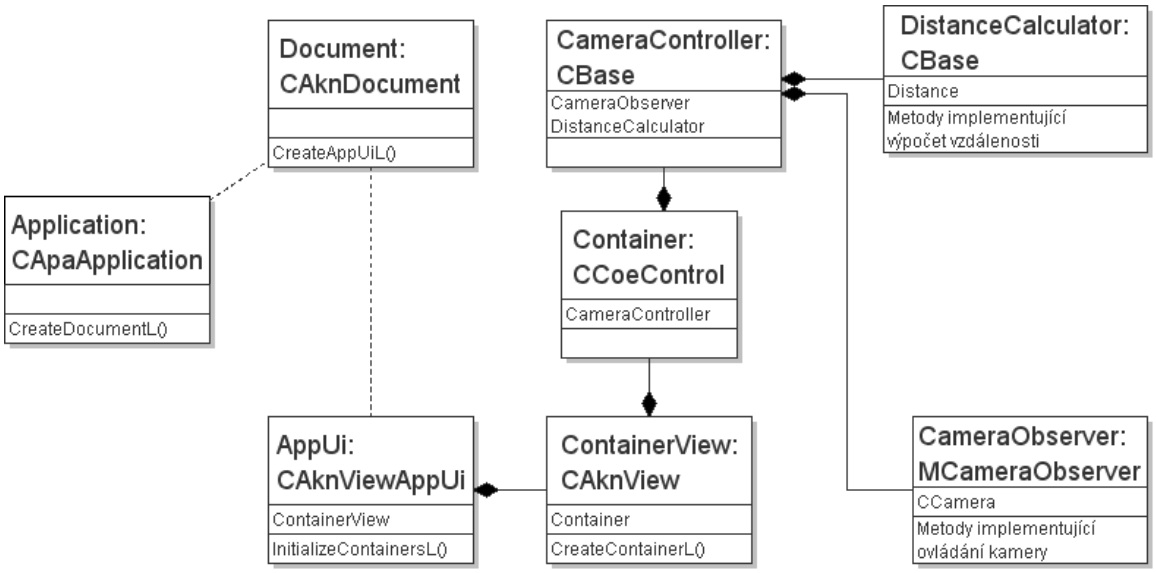
Aplikace by měla umožnit kalibraci pro měření ve skutečných měrných jednotkách (centimetry) a také přepínání mezi měřením absolutní dráhy (tj. celkem uražené) a relativní vzdálenosti (vzdálenost od bodu počátku měření). Uživatelské rozhraní by mělo být co nejjednodušší a snadno použitelné.

## 4.2 Implementace

Během vývoje byl jako referenční použit mobilní telefon Nokia 6120 Classic, což je zařízení s operačním systémem Symbian S60 3rd Edition Feature Pack 1, tedy Symbian 9.2. Přístroj pohání CPU ARM 11 o frekvenci 369MHz a má 64MB RAM, přičemž pro aplikace je volných přibližně 20MB. Programoval jsem v integrovaném vývojovém prostředí Nokia Carbide.c++.

Aplikace je implementována jako standardní symbianová S60 aplikace s uživatelským rozhraním s podporou pohledů. Pohled je implementován pouze jeden, ale tato architektura umožňuje budoucí rozšiřitelnost.

Třída, která obsluhuje integrovanou kameru, definuje virtuální metody třídy `MCameraObserver`. Snímání jednotlivých snímků je prováděno pomocí zachytávání videa do paměti, takže pro spouštění metod pro výpočet vzdálenosti je použita metoda observeru `FrameBufferReady()`, která na vstupu dostane snímek ve formátu YUV420. Vzhledem k tomu, že kamera dodává snímky po jednom, začne třída pro výpočet vzdálenosti počítat až po dodání druhého snímku.




Obrázek 4.1: Diagram tříd výsledné aplikace.

Pro výpočet se používají obrazy v odstínech šedi, díky použití formátu YUV není nutné dodané snímky nijak upravovat. Jednotlivé snímky mají nejnižší možné rozlišení jaké daný přístroj nabízí, u referenčního telefonu je to 128x96 pixelů. Rozlišení snímků na přenost algoritmu nemá vliv, ale ovlivňuje výpočetní nároky, což přesnost ovlivnit může, jak uvidíme dále. Frekvence snímkování je nastavena na co nejvyšší hodnotu, u referenčního přístroje 15fps.

Výpočet začíná určením parciálních derivací intenzity (úrovně šedi) a pokračuje výpočtem soustavy rovnic uvedených v kapitole 2. Výsledkem jsou hodnoty $u$ a $v$, tedy $x$-ová a $y$-ová komponenta optického toku, což je v podstatě vzdálenost na ose $x$ a $y$, o kterou se obraz posunul mezi prvním a druhým snímkem. Celkovou vzdálenost vypočítáme jednoduše Pythagorovou větou.

Zde se však dostáváme ke dvěma problémům. Prvním z nich je omezený výpočetní výkon mobilního telefonu a druhým nízký počet snímků za sekundu, který je integrovaný fotoaparát schopen dodávat. Nízký výpočetní výkon se projeví v okamžiku, kdy počítáme parciální derivace intenzity pro každý bod obrazu. Počet parciálních derivací je bezmála roven trojnásobku počtu pixelů ve snímku (parciální derivace podle tří os), jejich výpočet sice lze optimalizovat pouze na jeden průchod oběma obrazy, ale i tak je výpočet vcelku náročný a referenční telefon ho není schopen provádět plynule ani při nevysoké frekvenci 15 snímků za vteřinu. Zmíněná frekvence je sice nejvyšší, jakou kamera telefonu nabízí, ale snižovat ji není možné, právě z důvodu druhého problému.

Pro určení optického toku je frekvence snímkování klíčová z toho důvodu, že musí být pohyb mezi jednotlivými snímky co nejmenší, abychom dosáhli co nejpřesnějších výsledků. Přesnost je nejvyšší, pokud je pohyb mezi snímky roven velikost jednoho obrazového bodu. Empiricky jsem ověřil, že pokud pro výpočet použiji každý pixel jednotlivých snímků, je nutné přístrojem, který dokáže snímat pouze 15 snímků za sekundu, pohybovat neúnosně pomalu, navíc jak již bylo řečeno výše, přístroj nemá dostatečný výkon pro plynulý výpočet, takže dochází k zahazování nepoužitých snímků, čímž se frekvence snímkování dále snižuje.

Oba uvedené problémy však mají společné řešení, které navíc dokáže omezit vliv šumu na přesnost výsledků. Tímto řešením je uměle snížit počet pixelů, při zachování poměru

k reálné vzdálenosti, a to tak, že za nové obrazové body budeme považovat oblasti o velikost několika pixelů ohodnocené jejich průměrnou intenzitou. Podobné řešení, ovšem pouze pro omezení šumu doporučuje i Horn[6] a také bylo použito v podobné aplikaci[14]. S rostoucí velikostí oblastí samozřejmě bude klesat přesnost, ale zároveň se zvyšovat rychlost, kterou můžeme telefonem pohybovat, a také poroste plynulost výpočtu. Aplikace proto umožňuje přepínání velikostí daných oblastí, a to mezi hodnotami 1, 4, 8 a 16 pixelů, přičemž za dobře použitelné, vzhledem k rychlosti pohybu i výpočtu, považuji hodnoty 4 a 8, při hodnotě 16 je přesnost už velmi nízká, aplikace je schopna zaznamenat pohyb pouze nad velmi různorodým povrchem a při hodnotě 1 je nutno přístrojem pohybovat opravdu velice pomalu.

Protože takto by aplikace pouze počítala o kolik pixelů, nebo čtverců o různych velikostech se obraz posunul, je nutné implementovat převod na skutečné jednotky, například centimetry. Aby bylo možno tento převod implemetovat, musíme programu nějakým způsobem sdělit, kolik pixelů je jeden centimetr, k tomu slouží kalibrace. Při kalibraci je uživatel vyzván, aby přístrojem vykonal pohyb v délce 30cm, hodnota naměřená při tomto pohybu se uloží a dále je použita k přepočtu dalších naměřených hodnot na reálné jednotky. K přepočtu je použit triviální vztah

$$\texttt{vzdálenost v cm} = \texttt{naměřeno} \cdot \frac{30}{\texttt{kalibrační hodnota}}$$

Při použití různých citlivostí jsou naměřené hodnoty sice různé, ale protože jsou vždy převedeny jednotlivé pixely, lze kalibraci provádět při jakémkoli nastavení citlivosti. Hodnota získaná kalibrací je samozřejmě uložena do FLASH paměti přístroje, aby nebylo nutné provádět nové nastavení při každém spuštění aplikace. Vzdálenost 30 cm nebyla zvolena náhodně, ale z důvodu možnosti kalibrace bez použití metru či jiného měřítka, 30 cm je totiž často používaná míra, pro určení přibližně stejného rozměru lze totiž s úspěchem použít list A4.

Pro návrh uživatelského prostředí byl použit vestavěný UI Designer vývojového prostředí Carbide.c++ a u tříd použitých pro ovládání kamery jsem se inspiroval ukázkovým programem od společnosti Nokia[12]. Uživatelská příručka popisující jednotlivé ovládací prvky je přílohou této práce.

# Kapitola 5

# Testování a výsledky

V této kapitole jsou popsány všechny prováděné testy, kromě samotné metodiky je také zdůvodněno použití právě takových nastavení. Hlavním obsahem je však shrnutí a zhodnocení výsledků provedených testů.

## 5.1 Základní testy

Základní testy spočívají v měření uražené vzdálenosti nad různými povrchy, a to ve výšce mezi 5 a 10 cm nad těmito povrchy.

### 5.1.1 Metodika testování

Základní testy byly prováděny na třech různých druzích povrchů, které jsou běžně dosažitelné v reálném prostředí. První z nich je barevně velmi členitý koberec, na kterém by přesnost měla být nejvyšší. Tento povrch ukazuje obrázek 5.1. Druhý test byl prováděn na podlahové krytině s texturou dřeva (obrázek 5.2). Poslední test byl prováděn na hrubě texturované tkanině (obrázek 5.3). Všechny tři fotografie byly vytvořeny testovacím mobilním telefonem v podmínkách testování a pro srovnání také běžným digitálním fotoapaáratem.

Všechna měření byla prováděna v deseti pokusech na každém povrchu ve třech různých nastaveních citlivosti. Vyšší citlivost s použitím oblastí představujících nové obrazové body (viz 4.2) o velikosti 4x4 pixelů, nižší s velikostí oblastí 8x8 pixely a nejnižší s oblastmi o velikost 16x16 pixelů. Nejvyšší citlivost, která nepoužívá uvedené oblasti vůbec, tedy je počítáno s hodnotami každého jednotlivého pixelu, testována nebyla. Její výpočetní nároky jsou totiž tak vysoké, že výpočet vzdálenosti uražené mezi dvěma následujícími snímky nebyl dokončen dříve, než kamera dodala nový snímek. Nový snímek byl proto zahozen, čímž došlo k snížení framerate a tím k nepřenosti. Rozhodnutí nepoužít nejvyšší nastavení citlivosti při testování podkládám tabulkou 5.1 a 5.2. Hodnoty v tabulce 5.2 ukazují vzdálenosti naměřené na prvním testovacím povrchu, jak je z výsledků vidět jsou tyto hodnoty naprosto nepřesvědčivé, a proto jsem se nejvyšší citlivost rozhodl pro další testování nepoužít.

Protože jsem měření prováděl různou rychlostí jsou v hodnotách vidět velké rozdíly, například měření č. 2 jsem prováděl skutečně velmi pomalu, celý pohyb dlouhý 30 cm trval téměř než 30 sekund a přesto naměřená hodnota není dostaečně přesná. Měření č. 4 jsem provedl rychlostí obvyklou pro měření s jiným nastavením citlivosti, jak je vidět, naměřená hodnota je naprosto mimo. U dalším měření se rychlost různě měnila, ale lze říci, že čím pomaleji jsem přístrojem pohyboval, tím lepší výsledek byl, přesto je kvalita výsledků naprosto nedostačující pro normální použití.

Digitální fotoaparát, rozlišení 2592x1944 px, zmenšeno.

Testovací mobilní telefon, rozl. 320x240 px, zvětšeno.

Obrázek 5.1: První testovací povrch.

Digitální fotoaparát, rozlišení 2592x1944 px, zmenšeno.

Testovací mobilní telefon, rozl. 320x240 px, zvětšeno.

Obrázek 5.2: Druhý testovací povrch.

Důvod, proč je při nejvyšším nastavení citlivosti nutné pohybovat přístrojem velice pomalu je vidět v tabulce 5.1. Protože výpočet vzdálenosti uražené mezi dvěma snímky trvá dlouho, dodá kamera další snímek dřív, než je výpočet dokončen. Nový snímek musí


Obrázek 5.3: Třetí testovací povrch.

být zahozen a dále počítáno až se snímkem následujícím. Jednotlivé obrazy proto na sebe přesně nenavazují, proto je nutné aby změny mezi jednotlivými i nenavazujícími snímky byly co nejmenší, aby nebyla ovlivněna přesnost výpočtu.

| Citlivost: | 16x16 | 8x8 | 4x4 | 1x1 |
|---|---|---|---|---|
| Průměrná snímk. frekvence | 15 | 15 | 15 | 5 |

Tabulka 5.1: Průměrná snímkovací frekvence při různém nastavení citlivosti

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 1x1 | 11 | 24 | 14 | 4 | 14 | 18 | 11 | 10 | 11 | 14 | |
| Odchylka [cm] | 19 | 6 | 16 | 26 | 16 | 12 | 19 | 20 | 19 | 16 | 16,9 cm |
| Odchylka [%] | 63,3 | 20 | 53,3 | 86,6 | 53,3 | 40 | 63,3 | 66,6 | 63,3 | 53,3 | 56,33 % |

Tabulka 5.2: Výsledky testů na prvním testovacím povrchu při použití citlivosti 1x1 px.

Kalibrace byla provedena na prvním povrchu, na kterém by přesnost měla být nejvyšší. Délka každého měření byla 30 cm. Měření bylo prováděno rukou, proto se mohly vyskytnout i značné nepravidelnosti v pohybu, což však testování jen přibližuje reálnému použití.

Co se týče výsledků, za dostačující jsou považovány ty, jejichž odchylka nepřekročí 10 %. Ačkoli je i tato nepřesnost už dost vysoká, je stále ještě vyhovující pro běžné orientační měření.

### 5.1.2 Výsledky základních testů

#### Testy na prvním povrchu

Jak vidíme z výsledků (tabulka 5.3), je přesnost na prvním druhu povrchu při použití obou vyšších citlivostí velice podobná. Průměrnou odchylku kolem 5 % lze považovat za dobrý výsledek. 10 % odchylka při použití nejnižší citlivosti je sice horší, ale stále dostačující výsledek.

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 16x16 | 28 | 33 | 27 | 28 | 29 | 27 | 25 | 25 | 28 | 27 | |
| Odchylka [cm] | 2 | 3 | 3 | 2 | 1 | 3 | 5 | 5 | 2 | 3 | 2,9 cm |
| Odchylka [%] | 6,66 | 10 | 10 | 6,66 | 3,33 | 10 | 16,6 | 16,6 | 6,66 | 10 | 9,66 % |
| Citlivost 8x8 | 29 | 30 | 28 | 33 | 29 | 32 | 34 | 29 | 29 | 28 | |
| Odchylka [cm] | 1 | 0 | 2 | 3 | 1 | 2 | 4 | 1 | 1 | 2 | 1,7 cm |
| Odchylka [%] | 3,33 | 0 | 6,66 | 10 | 3,33 | 6,66 | 13,3 | 3,33 | 3,33 | 6,66 | 5,66 % |
| Citlivost 4x4 | 28 | 27 | 30 | 31 | 32 | 30 | 32 | 31 | 29 | 31 | |
| Odchylka [cm] | 2 | 3 | 0 | 1 | 2 | 0 | 2 | 1 | 1 | 2 | 1,4 cm |
| Odchylka [%] | 6,66 | 10 | 0 | 3,33 | 6,66 | 0 | 6,66 | 3,33 | 3,33 | 6,66 | 4,66 % |

Tabulka 5.3: Výsledky testů na prvním testovacím povrchu.

#### Testy na druhém povrchu

Tabulka 5.4 ukazuje, že v tomto testu je přesnost u vyšších citlivostí dokonce opačně k nastavení citlivosti, celková přesnost se však na méně gradientním povrchu očekávatelně snížila. Její hodnota je však stále nedosahuje 10 %, tudíž jde o vcelku dobrý výsledek. Přesnost na nejnižší citlivost se proti prvnímu povrchu snížila velice razantně, to je pochopitelné, protože detaily druhého testovacího povrchu jsou příliš jemné, než aby byly zachyceny v hrubém rastru 16x16 px.

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 16x16 | 15 | 13 | 14 | 16 | 17 | 22 | 15 | 18 | 18 | 17 | |
| Odchylka [cm] | 15 | 17 | 16 | 14 | 13 | 8 | 15 | 12 | 12 | 13 | 13,5 cm |
| Odchylka [%] | 50 | 56,6 | 53,3 | 46,6 | 43,3 | 26,6 | 50 | 40 | 40 | 43,3 | 45 % |
| Citlivost 8x8 | 25 | 30 | 27 | 32 | 30 | 28 | 26 | 27 | 26 | 29 | |
| Odchylka [cm] | 5 | 0 | 3 | 2 | 0 | 2 | 4 | 3 | 4 | 1 | 2,4 cm |
| Odchylka [%] | 16,6 | 0 | 10 | 6,66 | 0 | 6,66 | 13,3 | 10 | 13,3 | 3,33 | 8 % |
| Citlivost 4x4 | 28 | 27 | 28 | 24 | 28 | 29 | 25 | 30 | 33 | 32 | |
| Odchylka [cm] | 2 | 3 | 2 | 6 | 2 | 1 | 5 | 0 | 3 | 2 | 2,6 cm |
| Odchylka [%] | 6,66 | 10 | 6,66 | 20 | 6,66 | 3,33 | 16,6 | 0 | 10 | 6,66 | 8,66 % |

Tabulka 5.4: Výsledky testů na druhém testovacím povrchu.

**Testy na třetím povrchu.**

Na třetím testovacím povrchu (výsledky viz tabulka 5.5) se naplno projevily rozdíly v nastavení citlivosti. Na citlivost 4x4 odchylka lehce přesahuje 10 %, při 8x8 se pohybuje okolo 5 %, lze tedy hovořit o uspokojivých výsledcích. Špatný výsledek měření při citlivosti 16x16, lze stejně jako pokles přesnosti mezi 4x4 a 8x8 vysvětlit opět velikostí detailů na tomto povrchu, které jsou malé a přitom je plocha vcelku pravidelně uspořádána. Jednotlivé oblasti o velikosti 16x16 pixelů proto mají téměř stejné hodnoty, plocha je tedy méně gradientní, čímž samozřejmě razantně klesá možnost určit optický tok správně.

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 16x16 | 13 | 15 | 12 | 15 | 10 | 13 | 11 | 13 | 14 | 15 | |
| Odchylka [cm] | 17 | 15 | 18 | 15 | 20 | 17 | 19 | 17 | 16 | 15 | 16,9 cm |
| Odchylka [%] | 56,6 | 50 | 60 | 50 | 66,6 | 56,6 | 63,3 | 56,6 | 53,3 | 50 | 56,33 % |
| Citlivost 8x8 | 34 | 30 | 30 | 35 | 22 | 30 | 29 | 24 | 34 | 27 | |
| Odchylka [cm] | 4 | 0 | 0 | 5 | 8 | 0 | 1 | 7 | 4 | 3 | 3,2 cm |
| Odchylka [%] | 13,3 | 0 | 0 | 16,6 | 26,6 | 0 | 3,33 | 23,3 | 13,3 | 10 | 10,66 % |
| Citlivost 4x4 | 34 | 28 | 29 | 32 | 32 | 27 | 31 | 30 | 29 | 31 | |
| Odchylka [cm] | 4 | 2 | 1 | 2 | 2 | 3 | 1 | 0 | 1 | 1 | 1,7 cm |
| Odchylka [%] | 13,3 | 6,66 | 3,33 | 6,66 | 6,66 | 10 | 3,33 | 0 | 3,33 | 3,33 | 5,66 % |

Tabulka 5.5: Výsledky testů na prvním testovacím povrchu.

## 5.2 Další testy

Další testy měli za cíl prozkoumat využití aplikace pro měření na větší vzdálenosti, než jen několik centimetrů od měřeného povrchu. Pro takové testy je samozřejmě nutné aplikaci jinak nakalibrovat, protože hodnoty v pixelech jsou při takových měřeních naprosto odlišné.

### 5.2.1 Metodika testování

Tyto testy byly provedeny dva a pouze s jedním nastavením citlivosti, a to 4x4 px. První test probíhal opět nad prvním testovacím povrchem, ale tentokrát jsem přístroj držel ve výšce pasu, tedy cca 80cm, měřena byla vzdálenost 60cm. Druhý test probíhal nad druhým testovacím povrchem, ve stejné výšce jako první, výška a měřená vzdálenost byly shodné s prvním testem. Kalibrace proběhla pro každý povrch zvlášť.

### 5.2.2 Výsledky testů

**Test na prvním povrchu.**

Průměrná odchylka při tomto testu (výsledky v tabulce 5.6) byla 5,1 cm, což je 8,5 %, tedy výsledek nižší než 10 %.

**Test na druhém povrchu.**

Jak můžeme vidět v tabulce 5.7, jsou výsledky tohoto testu překvapivě jen mírně horší než výsledky testu na prvním povrchu. Druhý povrch je málo gradientní i z blízka a z výšky se

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 4x4 | 60 | 62 | 67 | 52 | 66 | 64 | 62 | 66 | 67 | 51 | |
| Odchylka [cm] | 0 | 2 | 7 | 8 | 6 | 4 | 2 | 6 | 7 | 9 | 5,1 cm |
| Odchylka [%] | 0 | 3,33 | 11,6 | 13,3 | 10 | 6,66 | 3,33 | 10 | 11,6 | 15 | 8,5 % |

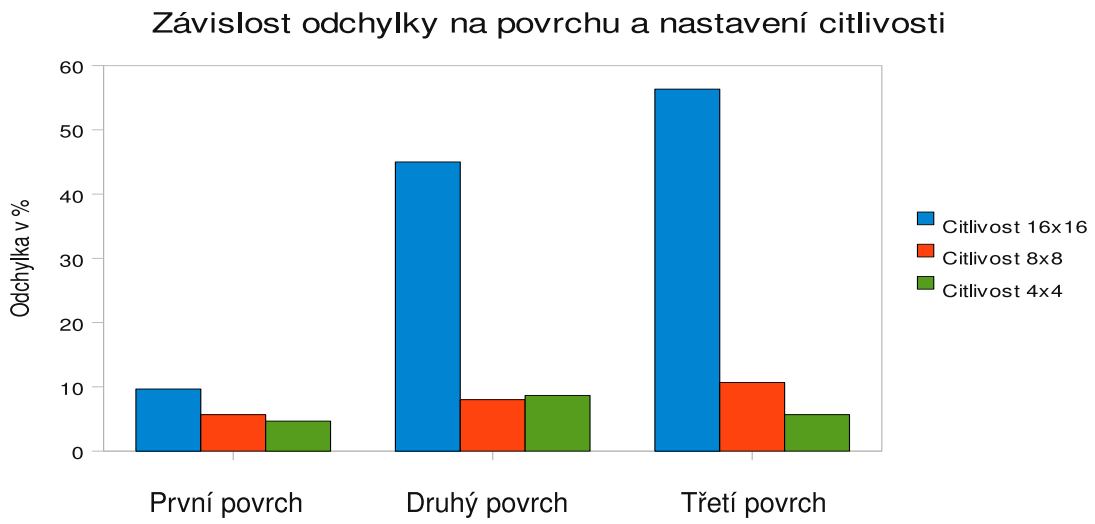
Tabulka 5.6: Výsledek druhého testu na prvním testovacím povrchu.

jeho detaily ještě více slévají, proto je průměrná odchylka v hodnotě necelých 10 % nečekaně nízká.

| | Naměřené hodnoty | | | | | | | | | | Prům. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Měření: | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | odch. |
| Citlivost 4x4 | 53 | 57 | 58 | 53 | 51 | 55 | 54 | 56 | 52 | 53 | |
| Odchylka [cm] | 7 | 3 | 2 | 7 | 9 | 5 | 6 | 4 | 8 | 7 | 5,8 cm |
| Odchylka [%] | 11,6 | 5 | 3,33 | 11,6 | 15 | 8,33 | 10 | 6,66 | 13,3 | 11,6 | 9,66 % |

Tabulka 5.7: Výsledek druhého testu na druhém testovacím povrchu.

## 5.3 Shrnutí testování




Obrázek 5.4: Graf závislosti přesnosti měření na nastavení citlivosti a měřeném povrchu.

Výsledky základních testů potvrdily výchozí předpoklad, že s klesající členitostí povrchu bude klesat i přesnost výsledků. Vliv nastavení citlivosti na přesnost výsledků na různých površích ukazuje graf 5.4. Při citlivosti 4x4 a 8x8 pixelů jsou výsledky na všech třech testovacích površích velice podobné, je tomu tak proto, že texturování těchto povrchů je dostatečně hrubé. Naopak pro měření na povrchu s jemnějšími detaily je aplikace nepoužitelná, protože drobné rozdíly nezachytí ani vyšší citlivost, jistou aproximaci měření na jemném povrchu poskytují výsledky měření na nejnižší citlivost. Vidíme, že na prvním, velmi gradientním

povrchu, je přesnost stále vcelku vysoká, ale na površích s jemnější texturou již o přesnosti nelze ani hovořit. Použití nejvyššího nastavení (bez průměrování hodnot větších oblastí) je v praxi nereálné, protože výpočet probíhá skutečně velmi pomalu.

Na výsledky má samozřejmě velký vliv vzdálenost přístroje od snímaného povrchu. Důvod je ten, že vzdálenější kamera zabírá větší oblast povrchu při zachování stejného rozlišení. Interní reprezentace vzdálenosti je však v pixelech, a proto přepočet neplatí. Měření je nutno provádět v takové vzdálenosti, v jaké byla aplikace nakalibrována.

Použití aplikace pro měření z větší vzdálenosti ukazují dva závěrečné testy. Jejich výsledky jsou sice horší než při měření zblízka, ale to lze vysvětlit nepravidelnostmi pohybu při takovém měření. Mírně překvapivý je výsledek posledního testu. Tento test, ač byl prováděn na jemném povrchu a navíc ještě ze vzdálenosti, v které drobné detaily zanikají, přinesl vcelku dobré výsledky. Důvody tohoto chování můžeme hledat v kalibraci aplikace speciálně pro tento povrch.

# Závěr

Hlavním cílem práce bylo nalézt způsob, jak naprogramovat aplikaci pro mobilní telefon, která je schopná měřit vzdálenost pomocí integrovaného fotoaparátu. Tato problematika nespočívá jen v nalezení vhodného algoritmu, ale je také nutno správně zvolit implementační platformu. Platforem pro programování mobilních aplikací je totiž vcelku velké množství a jejich možnosti jsou značně rozdílné. Tato práce nenavazuje na žádné podobné projekty.

V úvodu práce jsou představeny různé metody, které lze využít pro měření pohybu v digitálním obraze. Následuje rozbor mobilních platforem s popisem jejich výhod a nevýhod. Na těchto teoretických základech jsem postavil druhou část práce, která se věnuje již samotnému návrhu a implementaci dané aplikace. Aplikace je naprogramována v jazyce C++ a funguje v operačním systému Symbian, což je jeden z nejrozšířenějších operačních systémů pro mobilní telefony.

V poslední kapitole práce jsou popsány testy aplikace a zhodnoceny jejich výsledky. Výsledky většinou odpovídají očekávání a neplynou z nich žádné překvapivé závěry. Jedinnou vyjímkou je poslední provedený test, tedy test z výšky 80 cm nad druhým testovacím povrchem, pokud si ale uvědomíme, že aplikace byla předem přímo pro tento povrch nakalibrována není ani tento výsledek nijak neočekávatelný.

Nejtěžší na celé práci bylo pochopit a zhodnotit jednotlivé metody detekce pohybu. Metoda, kterou jsem nakonec použil, tedy detekce konstantního optického toku, se mi jevila jako nejvhodnější pro řešení daného problému. Určité komplikace přineslo i samotné programování aplikace, jazyk C++ používaný pro tvorbu symbianových aplikací je v několika detailech rozdílný od standardního C++ a také celková filozofie je od vývoje pro PC mírně odlišná.

Využít by se výsledná aplikace dala jako improvizovaný měřící nástroj, ovšem pouze v takových případech, kdy je její 90 %–95 % přesnost dostačující. Větší využití, ovšem ne přímo této aplikace, spíše algoritmu pro měření pohybu, bych viděl například v intuitivním ovládání mobilního telefonu, kdy by se určitým pohybem daly spouštět různé akce. Také by neměl být problém v použití jednoduchých gest. Například již existují hry, které využívají podobný algoritmus pro ovládání zaměřovače střelby. Jednu z takových her jsem testoval[2] a subjektivně mohu říci, že přesnost detekce pohybu je u mé aplikace mnohem vyšší. Otázkou však je, zda by hardware mobilního telefonu měl dostatečný výkon pro výpočet mnou použitého algoritmu a ještě pro vykreslování herní grafiky.

Co se týče směřování dalšího vývoje, jednou z možností by bylo zaměřit se na zvýšení přesnosti měření. Nejsem si však jistý, zda toho jde dosáhnout pouhou úpravou či změnou algoritmu, nebo zda je výrazně vyšší přesnost mimo současné technické možnosti zařízení. Za druhý směr vývoje by se dalo považovat využití již naprogramovaného algoritmu v jiných aplikacích, například právě pro intuitivní ovládání přístroje.

# Literatura


[1] BARRON, J. L.; FLEET, D. J.; BEAUCHEMIN, S. S.: *Performace of Optical Flow Techniques*. [online], Naposledy navštíveno 20.4. 2008.
URL <http://www.cs.toronto.edu/~fleet/research/Papers/ijcv-94.pdf>

[2] C4MProd: *Nike: T90*. informace [online], Naposledy navštíveno 4.5. 2008.
URL <http://www.c4mprod.com/games.php?gameid=6>

[3] HARRISON, R.: *Programujeme aplikace pro Symbian OS v jazyce C++*. Brno: Computer Press, 2006, iSBN 80-251-1243-8.

[4] HLAVÁČ, V.; ŠONKA, M.: *Počítačové vidění*. Praha: Grada, 1992, iSBN 80-85424-67-3.

[5] HLAVÁČ, V.; ŠONKA, M.; BOYLE, R.: *Image Processing: Analysis and Machine Vision*. Kapitola 15 [online], Naposledy navštíveno 5.5. 2008.
URL <http://cmp.felk.cvut.cz/~hlavac/Public/TeachingText/ch15d.pdf>

[6] HORN, B. K. P.: *Determining Constant Optical Flow*. [online], Naposledy navštíveno 20.4. 2008.
URL <http://people.csail.mit.edu/bkph/articles/Fixed_Flow.pdf>

[7] HORN, B. K. P.; SCHUNCK, B. G.: *Determining Optical Flow*. [online], Naposledy navštíveno 20.4. 2008.
URL <http://people.csail.mit.edu/bkph/papers/Optical_Flow_OPT.pdf>

[8] HRNČÍŘ, Z.: *Optický tok v obrazových datech živých buňek*. Diplomová práce, Masarykova univerzita. Fakulta informatiky, Brno, 2006, vedoucí práce: Mgr. Vladimír Ulman.

[9] KANĚČKA, P.: *Vyhledání význačných bodů v rastrovém obraze*. Diplomová práce, Vysoké učení technické. Fakulta informačních technologií, Brno, 2007, vedoucí práce: Ing. Adam Herout, Ph.D.

[10] LACKO, L.: *Programujeme mobilní aplikace ve Visual Studiu .NET*. Brno: Computer Press, 2004, iSBN 80-251-0176-2.

[11] Nokia: *Forum Nokia*. [online], Naposledy navštíveno 1.5. 2008.
URL <http://forum.nokia.com/>

[12] Nokia: *S60 Platform: Camera Example with AutoFocus Support*. [online], Použita verze 2.0 z 21.2. 2008, v současnoti k dipozici pouze verze 2.2 z 9.4. 2008.
URL <http://sw.nokia.com/id/2f492479-ac8c-4c3e-aa90-cc883e190d83/S60_Platform_Camera_Example_with_AutoFocus_Support_v2_2_en.zip>

[13] Nokia Research Center: *Python for S60*. [online], Naposledy navštíveno 1.5. 2008.
URL <http://opensource.nokia.com/projects/pythonfors60/index.html>

[14] ROHS, M.; ESSL, G.: *CaMus*$^2$ *Optical Flow and Collaboration in Camera Phone Music Performance*. informace [online], Naposledy navštíveno 5.5. 2008.
URL <http://itp.nyu.edu/nime/2007/proc/nime2007_160.pdf>

[15] Sun Microsystems: *CDC: Java platform technology for connected devices*. [online], Naposledy navštíveno 1.5. 2008.
URL <http://java.sun.com/products/cdc/wp/cdc-whitepaper.pdf>

[16] Sun Microsystems: *Mobile Information Device Profile Datasheet*. [online], Naposledy navštíveno 1.5. 2008.
URL <http://java.sun.com/products/midp/midp-ds.pdf>

[17] Wikipedia: *Symbian OS*. [online], Naposledy navštíveno 1.5. 2008.
URL <http://en.wikipedia.org/wiki/Symbian_OS>

[18] Wikipedia: *Corner detection*. [online], Naposledy navštíveno 20.4. 2008.
URL <http://en.wikipedia.org/wiki/Corner_detection>

[19] Wikipedia: *Image sensor*. [online], Naposledy navštíveno 20.4. 2008.
URL <http://en.wikipedia.org/wiki/Image_sensor>

[20] Wikipedia: *Windows CE*. [online], Naposledy navštíveno 2.5. 2008.
URL <http://en.wikipedia.org/wiki/Windows_CE>

[21] Wikipedia: *Windows Mobile*. [online], Naposledy navštíveno 2.5. 2008.
URL <http://en.wikipedia.org/wiki/Windows_Mobile>

# Seznam příloh

## Příloha 1:

Uživatelská příručka aplikace

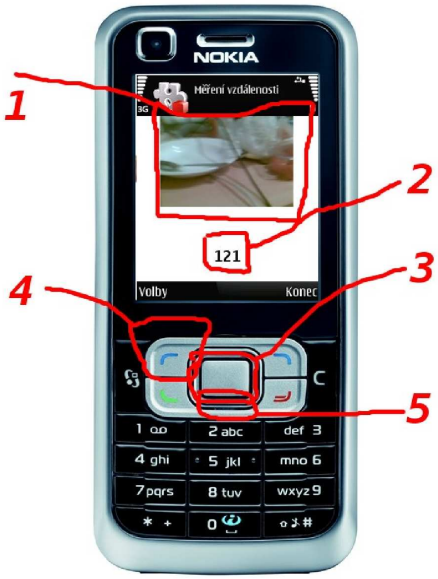
## Příloha 2:

CD s hotovou aplikací, zdrojovými kódy, programovou dokumentací, textem práce a uživatelskou příručkou.

# Uživatelská příručka

## *Aplikace pro měření vzdálenosti*

Základní ovládací prvky




1. Místo, kde se zobrazuje hledáček fotoaparátu.
2. Naměřená vzdálenost.
3. Tlačítkem „OK“ se zapíná a vypíná měření vzdálenosti.
4. Levé kontextové tlačítko zobrazuje menu.
5. Klávesou „Dolů“ se vynuluje naměřená vzdálenost.

Položka „Vynulovat“ vynuluje naměřenou vzdálenost, podobně jako klávesa „Dolů“.

Položkou „Kalibrace“ se spustí kalibrace.

Po zmizení dialogu, který vyzývá k vykonání 30cm dlouhého pohybu, tento pohyb vykonejte a potrvďte tlačítkem „OK“.


Položka „Citlivost“ umožňuje výběr ze čtyř nastavení citlivosti.

Položka „Druh měření“ umožňuje přepínání mezi měřením absolutní vzdálenosti a relativní vzdálenosti od bodu počátku měření.

Položka „Konec“ ukončí aplikaci.